Python: module genutil.salstat

genutil.salstat

index

This module has been written specifically for the SalStat statistics package.
It is an object oriented (and more limited) version of Gary Strangmans
stats.y module, and much code has been taken from there. The classes and
methods are usable from the command line, and some may prefer the OO style
to stats.py's functional style.

Modules

MA	RandomArray	cdms
MV	genutil.array_indexing	

Functions

ChiSquare(x, y, axis=0, df=1)

```
This method performs a chi square on 2 data set.
Usage: chisq, df, prob = ChiSquare(x,y)
Options:
    axisoptions 'x' | 'y' | 'z' | 't' | '(dimension_name)' | 0 | 1
    default value = 0. You can pass the name of the dimension or i
    (integer value 0...n) over which you want to compute the stati
    you can also pass 'xy' to work on both axes at once
    nperm is the number of permutation wanted, default len(axis)+1
```

ChiSquareVariance(x, y, axis=0, df=1)

```
This method performs a Chi Square test for the variance ratio.
Usage:
   chisquare, prob, [df] = ChiSquareVariance(data, usermean, axis=
Returns: chisquare, prob, [df] =
Options:
    axisoptions 'x' | 'y' | 'z' | 't' | '(dimension_name)' | 0 | 1
    default value = 0. You can pass the name of the dimension or i
    (integer value 0...n) over which you want to compute the stati
    you can also pass 'xy' to work on both axes at once
```

CochranesQ(*inlist, **kw)

```
This method performs a Cochrances Q test upon a list of lists.
Usage: q, df, prob = CochranesQ(*inlist)
inlist, being as many arrays as you  wish
Options:
    axisoptions 'x' | 'y' | 'z' | 't' | '(dimension_name)' | 0 | 1
    default value = 0. You can pass the name of the dimension or i
    (integer value 0...n) over which you want to compute the stati
    you can also pass 'xy' to work on both axes at once
    df=1 : if 1 then degrees of freedom are retuned

    WARNING: axis and df MUST be passed as keyword, as all argumen
```

FTest(data1, data2, uservar, axis=0, df=1)

```
This method performs a F test for variance ratio and needs a user
hypothesised variance to be supplied.
Usage: f, prob [,df1, df2] = FTest(data1, data2, uservar, axis=axi
Options:
    axisoptions 'x' | 'y' | 'z' | 't' | '(dimension_name)' | 0 | 1
    default value = 0. You can pass the name of the dimension or i
    (integer value 0...n) over which you want to compute the stati
    you can also pass 'xy' to work on both axes at once

    df =0: if set to 1 returns degrees of freedom
```

FriedmanChiSquare(*inlist, **kw)

```
This method performs a Friedman chi square (like a nonparametric
within subjects anova) on a list of lists.
Usage: sumranks, chisq, df, prob = FriedmanChiSqure(*args, axis=ax
inlist, being as many arrays as you  wish
Options:
    axisoptions 'x' | 'y' | 'z' | 't' | '(dimension_name)' | 0 | 1
    default value = 0. You can pass the name of the dimension or i
    (integer value 0...n) over which you want to compute the stati
    you can also pass 'xy' to work on both axes at once
    df=1 : if 1 then degrees of freedom are retuned

    WARNING: axis and df MUST be passed as keyword, as all argumen
```

KendallsTau(x, y, axis=0)

```
This method performs a Kendalls tau correlation upon 2 data sets.
Usage: tau, z, prob = KendallsTau(data1, data2)
Options:
    axisoptions 'x' | 'y' | 'z' | 't' | '(dimension_name)' | 0 | 1
    default value = 0. You can pass the name of the dimension or i
    (integer value 0...n) over which you want to compute the stati
    you can also pass 'xy' to work on both axes at once
```

KolmogorovSmirnov(x, y, axis=0)

```
This method performs a Kolmogorov-Smirnov test for unmatched sampl
upon 2 data vectors.
Usage: ks, prob = KolmogorovSmirnov(data1, data2)
```

```
Options:
    axisoptions 'x' | 'y' | 'z' | 't' | '(dimension_name)' | 0 | 1
    default value = 0. You can pass the name of the dimension or i
    (integer value 0...n) over which you want to compute the stati
    you can also pass 'xy' to work on both axes at once
```

KruskalWallisH(*inlist, **kw)

```
This method performs a Kruskal Wallis test (like a nonparametric
between subjects anova) on a serie of arrays.
Usage: h, df, prob = KruskalWallisH(*args,axis=axispoptions, df=1)
inlist, being as many arrays as you  wish
Options:
    axisoptions 'x' | 'y' | 'z' | 't' | '(dimension_name)' | 0 | 1
    default value = 0. You can pass the name of the dimension or i
    (integer value 0...n) over which you want to compute the stati
    you can also pass 'xy' to work on both axes at once
    df=1 : if 1 then degrees of freedom are retuned

    WARNING: axis and df MUST be passed as keyword, as all argumen
```

LinearRegression(x, y, df=1, axis=0)

```
This method performs a linear regression upon 2 data vectors.
Usage:  r, t, prob, slope, intercept, sterrest [,df] = LinearRegre
Options:
    axisoptions 'x' | 'y' | 'z' | 't' | '(dimension_name)' | 0 | 1
    default value = 0. You can pass the name of the dimension or i
    (integer value 0...n) over which you want to compute the stati
    you can also pass 'xy' to work on both axes at once

    df=1: If set to 1 then df is returned
```

MannWhitneyU(x, y, Z_MAX=6.0, axis=0)

```
This method performs a Mann Whitney U test for unmatched samples o
2 data vectors.
Usage: bigu, smallu, z, prob = MannWhitneyU(data1, data2, Z_MAX=6.
Options:
    axisoptions 'x' | 'y' | 'z' | 't' | '(dimension_name)' | 0 | 1
    default value = 0. You can pass the name of the dimension or i
    (integer value 0...n) over which you want to compute the stati
    you can also pass 'xy' to work on both axes at once

    Z_MAX: Maximum meaningfull value  for z probability (default =
```

OneSampleSignTest(x, y, axis=0)

```
OneSampleSignTest
This method performs a single factor sign test. The data must be
supplied to this method along with a user hypothesised mean value.
Usage:
nplus, nminus, z, prob = OneSampleSignTest(data, usermean, axis=ax
Returns: nplus, nminus, z, prob.
Options:
    axisoptions 'x' | 'y' | 'z' | 't' | '(dimension_name)' | 0 | 1
```

```
    default value = 0. You can pass the name of the dimension or i
    (integer value 0...n) over which you want to compute the stati
    you can also pass 'xy' to work on both axes at once
```

OneSampleTTest(x, y, axis=0, df=1)

```
This performs a single factor t test for a set of data and a user
hypothesised mean value.
Usage: t, prob [,df] = OneSampleTTest(data, usermean, axis=axisopt
Options:
    axisoptions 'x' | 'y' | 'z' | 't' | '(dimension_name)' | 0 | 1
    default value = 0. You can pass the name of the dimension or i
    (integer value 0...n) over which you want to compute the stati
    you can also pass 'xy' to work on both axes at once

    df=1 : If set to 1 then the degrees of freedom are returned
```

PairedPermutation(x, y, nperm=None, axis=0)

```
This method performs a permutation test for matched samples upon 2
 This code was modified from Segal and further modifed by C. Doutr
Usage: utail, crit, prob = PairedPermutation(x,y,nperm=None)
Options:
    axisoptions 'x' | 'y' | 'z' | 't' | '(dimension_name)' | 0 | 1
    default value = 0. You can pass the name of the dimension or i
    (integer value 0...n) over which you want to compute the stati
    you can also pass 'xy' to work on both axes at once
    nperm is the number of permutation wanted, default len(axis)+1
```

PearsonsCorrelation(x, y, axis=0, df=1)

```
This method performs a Pearsons correlation upon two sets of data
Usage: r, t, prob, [df] = PearsonsCorrelation(data1, data2,axis=0,
Options:
    axisoptions 'x' | 'y' | 'z' | 't' | '(dimension_name)' | 0 | 1
    default value = 0. You can pass the name of the dimension or i
    (integer value 0...n) over which you want to compute the stati
    you can also pass 'xy' to work on both axes at once

    df =0: if set to 1 returns degrees of freedom
```

Range(x, axis=0)

```
Returns the range of the data
Usage:
    rg=Range(data,axis=axisoptions)
Options:
    axisoptions 'x' | 'y' | 'z' | 't' | '(dimension_name)' | 0 | 1
    default value = 0. You can pass the name of the dimension or i
    (integer value 0...n) over which you want to compute the stati
    you can also pass 'xy' to work on both axes at once
```

SpearmansCorrelation(x, y, axis=0, df=1)

```
This method performs a Spearmans correlation upon 2 data sets
Usage: rho, t, df, prob = SpearmansCorrelation(data1, data2, axis=
Options:
```

```
axisoptions 'x' | 'y' | 'z' | 't' | '(dimension_name)' | 0 | 1
default value = 0. You can pass the name of the dimension or i
(integer value 0...n) over which you want to compute the stati
you can also pass 'xy' to work on both axes at once

df=1 : If set to 1 returns the degrees of freedom
```

TTestPaired(x, y, axis=0, df=1)

```
This performs an paired t-test.
Usage: t, prob, [df] = TTestUnpaired(data1, data2,axis=axisoptions
Options:
    axisoptions 'x' | 'y' | 'z' | 't' | '(dimension_name)' | 0 | 1
    default value = 0. You can pass the name of the dimension or i
    (integer value 0...n) over which you want to compute the stati
    you can also pass 'xy' to work on both axes at once

    df =0: if set to 1 returns degrees of freedom
```

TTestUnpaired(x, y, axis=0, df=1)

```
This performs an unpaired t-test.
Usage: t, prob, [df] = TTestUnpaired(data1, data2,axis=axisoptions
Returns: t, df, prob
Options:
    axisoptions 'x' | 'y' | 'z' | 't' | '(dimension_name)' | 0 | 1
    default value = 0. You can pass the name of the dimension or i
    (integer value 0...n) over which you want to compute the stati
    you can also pass 'xy' to work on both axes at once

    df =0: if set to 1 returns degrees of freedom
```

TwoSampleSignTest(x, y, axis=0)

```
This method performs a 2 sample sign test for matched samples on 2
supplied data sets
Usage: nplus, nminus, ntotal, z, prob = TwoSampleSignTest(data1, d
Options:
    axisoptions 'x' | 'y' | 'z' | 't' | '(dimension_name)' | 0 | 1
    default value = 0. You can pass the name of the dimension or i
    (integer value 0...n) over which you want to compute the stati
    you can also pass 'xy' to work on both axes at once
```

WilcoxonRankSums(x, y, Z_MAX=6.0, axis=0)

```
This method performs a Wilcoxon rank sums test for unpaired design
upon 2 data vectors.
Usage: z, prob = WilcoxonRankSums(data1, data2, Z_MAX = 6.0, axis=
Options:
    axisoptions 'x' | 'y' | 'z' | 't' | '(dimension_name)' | 0 | 1
    default value = 0. You can pass the name of the dimension or i
    (integer value 0...n) over which you want to compute the stati
    you can also pass 'xy' to work on both axes at once

    Z_MAX: Maximum meaningfull value  for z probability (default =
```

WilcoxonSignedRanks(x, y, Z_MAX=6.0, axis=0)

```
This method performs a Wilcoxon Signed Ranks test for matched samp
upon 2 data set.
Usage: wt, z, prob = WilcoxonSignedRanks(data1, data2, Z_MAX = 6.0
Options:
    axisoptions 'x' | 'y' | 'z' | 't' | '(dimension_name)' | 0 | 1
    default value = 0. You can pass the name of the dimension or i
    (integer value 0...n) over which you want to compute the stati
    you can also pass 'xy' to work on both axes at once

    Z_MAX: Maximum meaningfull value  for z probability (default =
```

anovaBetween(*inlist, **kw)

```
This method performs a univariate single factor between-subjects
analysis of variance on a list of lists (or a Numeric matrix). It
specialised for SalStat and best left alone.
Usage: SSbet, SSwit, SStot, MSbet, MSerr, F, prob [, dfbet, dferr,
inlist, being as many arrays as you  wish
Options:
    axisoptions 'x' | 'y' | 'z' | 't' | '(dimension_name)' | 0 | 1
    default value = 0. You can pass the name of the dimension or i
    (integer value 0...n) over which you want to compute the stati
    you can also pass 'xy' to work on both axes at once
    df=1 : if 1 then degrees of freedom are retuned

    WARNING: axis and df MUST be passed as keyword, as all argumen
```

anovaWithin(*inlist, **kw)

```
This method is specialised for SalStat, and is best left alone.
For the brave:
Usage:
SSint, SSres, SSbet, SStot, MSbet, MSwit, MSres, F, prob [, dfbet,
inlist, being as many arrays as you  wish
Options:
    axisoptions 'x' | 'y' | 'z' | 't' | '(dimension_name)' | 0 | 1
    default value = 0. You can pass the name of the dimension or i
    (integer value 0...n) over which you want to compute the stati
    you can also pass 'xy' to work on both axes at once
    df=1 : if 1 then degrees of freedom are retuned

    WARNING: axis and df MUST be passed as keyword, as all argumen
```

betacf(a, b, x, ITMAX=200, EPS=2.9999999999999999e−07)

```
This function evaluates the continued fraction form of the incompl
Beta function, betai.  (Adapted from: MAal Recipies in C.)

Usage:   beta = betacf(a,b,x,ITMAX=200,EPS=3.0E-7)
ITMAX: Maximum number of iteration
EPS: Epsilon number
```

betai(a, b, x, ITMAX=200, EPS=2.9999999999999999e−07)

```
Returns the incomplete beta function:

I-sub-x(a,b) = 1/B(a,b)*(Integral(0,x) of t^(a-1)(1-t)^(b-1) dt)

where a,b>0 and B(a,b) = G(a)*G(b)/(G(a+b)) where G(a) is the gamm
function of a.  The continued fraction formulation is implemented
using the betacf function.  (Adapted from: MAal Recipies in C.)

Usage:  beta = betai(a,b,x,ITMAX=200,EPS=3.0E-7)
ITMAX: Maximum number of iteration for betacf
EPS: Epsilon number
```

center(x, axis=0)

```
Returns the deviation from the mean
Usage:
centered=center(data) # returns deviation from mean
Options:
    axisoptions 'x' | 'y' | 'z' | 't' | '(dimension_name)' | 0 | 1
    default value = 0. You can pass the name of the dimension or i
    (integer value 0...n) over which you want to compute the stati
    you can also pass 'xy' to work on both axes at once
```

chisqprob(chisq, df, Z_MAX=6.0)

```
Returns the (1-tailed) probability value associated with the provi
chi-square value and df.  Adapted from chisq.c in Gary Perlman's |

Usage:   prob = chisqprob(chisq,df)
Options:
Z_MAX: Maximum meaningfull value for z probability (default=6.0)
```

coefficentvariance(x, axis=0)

```
Returns the coefficents variance of data
Usage:
coefvar=coefficentvariance(data,axis=axisoptions)
Options:
    axisoptions 'x' | 'y' | 'z' | 't' | '(dimension_name)' | 0 | 1
    default value = 0. You can pass the name of the dimension or i
    (integer value 0...n) over which you want to compute the stati
    you can also pass 'xy' to work on both axes at once
```

differencesquared(x, y, axis=0)

```
Computes the Squared differecne between 2 datasets
Usage:
    diff=differencesquared(a,b)
Options:
    axisoptions 'x' | 'y' | 'z' | 't' | '(dimension_name)' | 0 | 1
    default value = 0. You can pass the name of the dimension or i
    (integer value 0...n) over which you want to compute the stati
    you can also pass 'xy' to work on both axes at once
```

erfcc(x)

```
Returns the complementary error function erfc(x) with fractional
error everywhere less than 1.2e-7.  Adapted from MAal Recipies.

Usage:   err = erfcc(x)
Options:
    axisoptions 'x' | 'y' | 'z' | 't' | '(dimension_name)' | 0 | 1
    default value = 0. You can pass the name of the dimension or i
    (integer value 0...n) over which you want to compute the stati
    you can also pass 'xy' to work on both axes at once
```

fprob(dfnum, dfden, F)

```
Returns the (1-tailed) significance level (p-value) of an F
statistic given the degrees of freedom for the numerator (dfR-dfF)
the degrees of freedom for the denominator (dfF).

Usage:   prob = fprob(dfnum, dfden, F)   where usually dfnum=dfbn,
```

gamma(x)

```
Returns the gamma function of x.
Gamma(z) = Integral(0,infinity) of t^(z-1)exp(-t) dt.
(Adapted from: MAal Recipies in C.)

Usage:   _gammaln(xx)
```

harmonicmean(x, axis=0)

```
Returns the harmonicmean of the data
Usage:
h=harmonicmean(data,axis=axisoptions)
Options:
    axisoptions 'x' | 'y' | 'z' | 't' | '(dimension_name)' | 0 | 1
    default value = 0. You can pass the name of the dimension or i
    (integer value 0...n) over which you want to compute the stati
    you can also pass 'xy' to work on both axes at once
```

inversechi(prob, df)

```
This function calculates the inverse of the chi square function. G
a p-value and a df, it should approximate the critical value neede
achieve these functions. Adapted from Gary Perlmans critchi functi
C. Apologies if this breaks copyright, but no copyright notice was
attached to the relevant file.
Usage invchi = inversechi(prob,df,axis=axisoptions)
```

inversef(prob, df1, df2)

```
This function returns the f value for a given probability and 2 gi
degrees of freedom. It is an approximation using the fprob functic
Adapted from Gary Perlmans critf function - apologies if copyright
broken, but no copyright notice was attached
Usage: fval = inversef(prob, df1, df2)
Options:
    axisoptions 'x' | 'y' | 'z' | 't' | '(dimension_name)' | 0 | 1
    default value = 0. You can pass the name of the dimension or i
    (integer value 0...n) over which you want to compute the stati
```

```
    you can also pass 'xy' to work on both axes at once
```

ksprob(x)

```
Computes a Kolmolgorov-Smirnov t-test significance level.  Adapted
MAal Recipies.

Usage:   ks = ksprob(x)
```

kurtosis(x, axis=0)

```
Return kurtosis value from dataset
Usage:
k=kurtosis(data, axis=axisoptions)
Options:
    axisoptions 'x' | 'y' | 'z' | 't' | '(dimension_name)' | 0 | 1
    default value = 0. You can pass the name of the dimension or i
    (integer value 0...n) over which you want to compute the stati
    you can also pass 'xy' to work on both axes at once
```

mad(x, axis=0)

```
return the sum of the deviation from the median
Usage:
md=_mad(data,axis=axisoptions)
Options:
    axisoptions 'x' | 'y' | 'z' | 't' | '(dimension_name)' | 0 | 1
    default value = 0. You can pass the name of the dimension or i
    (integer value 0...n) over which you want to compute the stati
    you can also pass 'xy' to work on both axes at once
```

median(x, axis=0)

```
Not really sophisticated median, based of arrays dimension,
Not to use with missing values
Usage:
med=_median(data,axis=axisoptions)
Options:
    axisoptions 'x' | 'y' | 'z' | 't' | '(dimension_name)' | 0 | 1
    default value = 0. You can pass the name of the dimension or i
    (integer value 0...n) over which you want to compute the stati
    you can also pass 'xy' to work on both axes at once
```

medianranks(x, axis=0)

```
Return the ranks of the median
Usage:
medrk=medianranks(data,axis=axisoptions)
Options:
    axisoptions 'x' | 'y' | 'z' | 't' | '(dimension_name)' | 0 | 1
    default value = 0. You can pass the name of the dimension or i
    (integer value 0...n) over which you want to compute the stati
    you can also pass 'xy' to work on both axes at once
```

mode(x, axis=0)

```
returns the mode of the data
Usage:
```

```
md=mode(data, axis=axisoptions)
Options:
    axisoptions 'x' | 'y' | 'z' | 't' | '(dimension_name)' | 0 | 1
    default value = 0. You can pass the name of the dimension or i
    (integer value 0...n) over which you want to compute the stati
    you can also pass 'xy' to work on both axes at once
```

numberuniques(x, axis=0)

```
Return the number of unique values
Usage:
uniques=numberuniques(data,axis=axisoptions)
Options:
    axisoptions 'x' | 'y' | 'z' | 't' | '(dimension_name)' | 0 | 1
    default value = 0. You can pass the name of the dimension or i
    (integer value 0...n) over which you want to compute the stati
    you can also pass 'xy' to work on both axes at once
```

pack(arrays)

```
Pack a list of arrays into one array
```

rankdata(x, axis=0)

```
Ranks the data, dealing with ties appropritely.
Adapted from Gary Perlman's |Stat ranksort.
Further adapted to MA/Numeric by PCMDI's team

Usage:   rankdata(array, axis=axisoptions)
Returns: a list of length equal to inlist, containing rank scores
Option:
    axisoptions 'x' | 'y' | 'z' | 't' | '(dimension_name)' | 0 | 1
        default value = 0. You can pass the name of the dimension
        (integer value 0...n) over which you want to compute the s
        even: 'xy': to do over 2 dimensions at once
```

skewness(x, axis=0)

```
Return the skewness of data
Usage:
skew=skewness(data, axis=axisoptions)
Options:
    axisoptions 'x' | 'y' | 'z' | 't' | '(dimension_name)' | 0 | 1
    default value = 0. You can pass the name of the dimension or i
    (integer value 0...n) over which you want to compute the stati
    you can also pass 'xy' to work on both axes at once
```

ssdevs(x, axis=0)

```
Return the sum of the square of the deviation from mean
Usage:
ss=_ssdevs(data,axis=axisoptions)
Options:
    axisoptions 'x' | 'y' | 'z' | 't' | '(dimension_name)' | 0 | 1
    default value = 0. You can pass the name of the dimension or i
    (integer value 0...n) over which you want to compute the stati
    you can also pass 'xy' to work on both axes at once
```

standarddeviation(x, axis=0)

```
Returns stadard deviation of data
Usage:
std=standarddeviation(data,axis=axisoptions)
Options:
    axisoptions 'x' | 'y' | 'z' | 't' | '(dimension_name)' | 0 | 1
    default value = 0. You can pass the name of the dimension or i
    (integer value 0...n) over which you want to compute the stati
    you can also pass 'xy' to work on both axes at once
```

standarderror(x, axis=0)

```
Returns the standard error from dataset
Usage:
stderr=standarderror(data,axis=axisoptions)
Options:
    axisoptions 'x' | 'y' | 'z' | 't' | '(dimension_name)' | 0 | 1
    default value = 0. You can pass the name of the dimension or i
    (integer value 0...n) over which you want to compute the stati
    you can also pass 'xy' to work on both axes at once
```

sumsquares(x, axis=0)

```
Return the sum of the squares
Usage:
    sq=sumsquare(data,axis=axisoptions)
Options:
    axisoptions 'x' | 'y' | 'z' | 't' | '(dimension_name)' | 0 | 1
    default value = 0. You can pass the name of the dimension or i
    (integer value 0...n) over which you want to compute the stati
    you can also pass 'xy' to work on both axes at once
```

tiecorrect(x, axis=0)

```
Corrects for ties in Mann Whitney U and Kruskal Wallis H tests.  S
Siegel, S. (1956) Nonparametric Statistics for the Behavioral Scie
New York: McGraw-Hill.  Code adapted from |Stat rankind.c code.

Usage:   T = tiecorrect(rankvals,axis=axisoptions)
Option:
    axisoptions 'x' | 'y' | 'z' | 't' | '(dimension_name)' | 0 | 1
        default value = 0. You can pass the name of the dimension
        (integer value 0...n) over which you want to compute the s
        even: 'xy': to do over 2 dimensions at once
```

tprob(df, t)

```
Returns t probabilty given degree of freedom and T statistic
Usage: prob = tprob(df,t)
```

unbiasedvariance(x, axis=0)

```
Return the variance (Ssq/(N-1))
Usage:
svar=unbiasedvariance(x,axis=axisoptions)
Options:
    axisoptions 'x' | 'y' | 'z' | 't' | '(dimension_name)' | 0 | 1
```

```
        default value = 0. You can pass the name of the dimension or i
        (integer value 0...n) over which you want to compute the stati
        you can also pass 'xy' to work on both axes at once
```

variance(x, axis=0)

```
Return the variance of data
Usage:
V=variance(data,axis=axisoptions)
Options:
    axisoptions 'x' | 'y' | 'z' | 't' | '(dimension_name)' | 0 | 1
    default value = 0. You can pass the name of the dimension or i
    (integer value 0...n) over which you want to compute the stati
    you can also pass 'xy' to work on both axes at once
```

zprob(z, Z_MAX=6.0)

```
Returns the area under the normal curve 'to the left of' the given
Thus,
for z<0, zprob(z) = 1-tail probability
for z>0, 1.0-zprob(z) = 1-tail probability
for any z, 2.0*(1.0-zprob(abs(z))) = 2-tail probability
Adapted from z.c in Gary Perlman's |Stat.

Z_MAX: Maximum meaningfull value  for z probability (default = 6)

Usage:   z = zprob(z,Z_MAX=6.0 )
```

Data

add = <MA.MA.masked_binary_operation instance>
multiply = <MA.MA.masked_binary_operation instance>